UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:
DISEÑO DE UNA WSN INCORPORANDO IPV6 PARA LA IMPLEMENTACIÓN DE UN PROTOTIPO DE SMART HOME ADMINISTRADO EN LA NUBE.

AUTORES:
DIEGO ARMANDO CORDERO CHANGO
EDISON MAURICIO PAUCAR CONDOR

TUTOR:
JUAN CARLOS DOMÍNGUEZ AYALA

Quito, febrero de 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros Diego Armando Cordero Chango y Edison Mauricio Paucar Condor, con documento de identificación N° 1721486882 y Nº 1723118913 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado:“ DISEÑO DE UNA WSN INCORPORANDO IPV6 PARA LA IMPLEMENTACIÓN DE UN PROTOTIPO DE SMART HOME ADMINISTRADO EN LA NUBE”, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.
En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en digital a la Biblioteca de la Universidad Politécnica Salesiana.

Nombre: Diego Armando Cordero Chango
Cédula: 1721486882

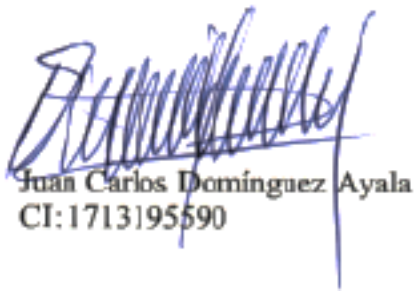
Nombre: Edison Mauricio Paucar Condor
Cédula: 1723118913

Quito, febrero 2020

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Artículo Académico, “DISEÑO DE UNA WSN INCORPORANDO IPV6 PARA LA IMPLEMENTACIÓN DE UN PROTOTIPO DE SMART HOME ADMINISTRADO EN LA NUBE”, realizado por Diego Armando Cordero Chango y Edison Mauricio Paucar Condor, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, febrero de 2020



DEDICATORIA

Dedico esta investigación con mucho cariño a:
A mis padres Lionso y Mónica por estar siempre conmigo en cada etapa de mi vida mostrándome su amor, trabajo, sacrificio, su preocupación y sobre todo su apoyo incondicional, a mi familia por sus consejos para no rendirme y que gracias a cada granito que han aportado he logrado culminar mi proceso académico. Gracias de corazón por ser la mejor familia.

Diego Cordero

A mi madre que cada día se esforzó por darme la educación y los recursos para llegar hasta aquí, a mi padre que me enseñó a hacer las cosas correctas hasta donde su vida le alcanzo y a todos quienes marcaron mi vida de una u otra manera.

Mauricio Paucar

Diseño de una WSN incorporando IPv6 para la implementación de un prototipo de Smart Home administrado en la nube

Juan Carlos Domínguez Ayala
Docente Universidad Politécnica Salesiana
Quito, Ecuador
jdominguez@ups.edu.ec

Cordero Chango Diego Armando
Ingeniería Electrónica
Universidad Politécnica Salesiana
Quito, Ecuador
dcorderoc@est.ups.edu.ec

Paucar Condor Edison Mauricio
Ingeniería Electrónica
Universidad Politécnica Salesiana
Quito, Ecuador
epaucarc@est.ups.edu.ec

***Resumen*— Se presenta el diseño de un prototipo de Smart Home mediante IoT ("Internet de las cosas") sobre la plataforma Re-Mote, de Zolertia consta de tres partes: la primera una WSN, como segunda parte un broker o servidor MQTT y la tercera parte un Sniffer. En la implementación de la red inalámbrica de sensores (WSN) se emplean cuatro motas Zolertia Re-mote, una que actúa como nodo Gateway junto con una Raspberry Pi 3 con scripts hechas en Python, dos que actúan como nodo sensores y una tercera como Sniffer. Las motas (nodo - sensor) envían datos de los sensores al Gateway en el estándar 6LowPan además del protocolo MQTT para la publicación de los mismos en una plataforma IoT (ubidots); en la implementación se analizan parámetros como su funcionalidad, latencia, TTL y cobertura.**

***Abstract*— The design of a Smart Home prototype is presented through IoT ("Internet of things") on the Zolertia Re-Mote platform, it consists of three parts: the first one WSN, as the second part a broker or MQTT server and the third part a sniffer. In the implementation of the wireless sensor network (WSN) four Zolertia Re-mote specks are used, one that acts as a Gateway node together with a Raspberry Pi 3 with scripts made in Python, two that act as a sensor node and a third as Sniffer. The spots (node - sensor) send sensor data to the Gateway in the 6LowPan standard in addition to the MQTT protocol for their publication on an IoT platform (ubidots); In the implementation parameters such as functionality, latency, TTL and coverage are analyzed.**

Keywords: 6LowPAN, WSN, Smarthome.

I. INTRODUCCIÓN

El IoT permite asignar a cualquier dispositivo una dirección IP, un ejemplo práctico de esto es la implementación de una Smart Home debido al decrecimiento de los costos de los equipos electrónicos, en la misma constantemente se monitorea y envía datos parametrizados a un usuario mediante un Smartphone o computador con conexión a internet [1].

Frente a este contexto se propone una comunicación directa entre una red WSN y una red IP convencional haciendo uso del estándar oficial de la IETF conocido como 6lowPan, lo que hará que cada nodo de la red tenga su propia dirección IPv6 para lograr una comunicación entre sí por medio de un router de frontera (Edge Router), sin embargo se hace uso del protocolo MQTT para la publicación de los datos recibidos por dichos nodos en un bróker local, para parametrizar ciertas acciones acorde a condiciones dentro de la red.

II. MARCO TEÓRICO

A. Protocolos IPv4, IPv6 y 6LowPan

En los últimos años los dispositivos con conexión a internet han tenido un crecimiento exponencial considerable, se estima que son más de 200 millones de dispositivos en consecuencia las direcciones IPv4 resultan insuficientes [2].

En la Tabla 1, se detallan las diferencias más relevantes entre los dos protocolos IPv4 e IPv6:

Sin embargo, como se puede identificar en la Tabla 1, el tamaño de la trama tanto de IPv4 como IPv6 es demasiado grande para que los microcontroladores generalmente destinados a la creación de una WSN la soporten puesto que funcionan con poco espacio de almacenamiento y escasa demanda de energía.

Para solucionar este problema se implementó 6LoWPAN (IPv6 Over Low Power Wireless Personal Area Networks) que básicamente permite que los paquetes IPv6 se puedan trasportar sobre IEEE 802.15.4, según el RFC 2560, añadiendo una

capa de adaptación en la pila de protocolos IP entre la capa de enlace y red, comprimiendo así el tamaño de la cabecera IP para que se ajuste al de una trama de IEEE 802.15.4, no obstante de no ser posible la compresión de un paquete procede a fragmentarlo para su envío [3].

Tabla 1. *Diferencia ente protocolos IPv4 e IPv6.*

Característica	IPv4	IPv6
Tamaño de dirección	32 bits	128 bits
Tamaño de red	8-30 bits	64 bits
Cabecera IP	20-60 bytes (variable)	40 bytes (fija)
Tamaño de los paquetes	576 Bytes requeridos	280 Bytes requeridos
Número de direcciones	2^{32} (limitado)	2^{128} (ilimitado)
Asignación de direcciones	Una dirección por host	Múltiples direcciones por Host
Tipos de direcciones	Unicast, Multicast y Broadcast	Unicast, Multicast y Anycast
Fragmentación de paquetes	Hosts y Routers	Solo en Hosts
Características de seguridad	IPsec adaptado	IPsec incorporado en el protocolo IPV6

En la Tabla 2 se muestra una comparación entre los modelos TCP/IP y 6LowPAN [4].

Tabla 2. *Comparación entre los modelos TCP/IP y 6LowPAN.*

Modelo TCP/IP				**Modelo 6LoWPAN**	
HTTP		RTP	**Aplicación**	Aplicación	
TCP	UDP	ICMP	**Transporte**	UDP	ICMP
IP			**Internet**	IPv6 / **adaptación LoWPAN**	
Ethernet MAC			**Enlace de datos**	IEEE 802.15.4 MAC	
Ethernet PHY			**Capa física**	IEEE 802.15.4 PHY	

B. Topologías de red

Al implementar una WSN se pueden usar tres clases de topologías: estrella, malla o híbrida.

Topología estrella: la transferencia de información se la hace a través de los nodos conectados directamente al nodo Gateway. En la Figura 1 se observa la topología en forma de estrella [5].

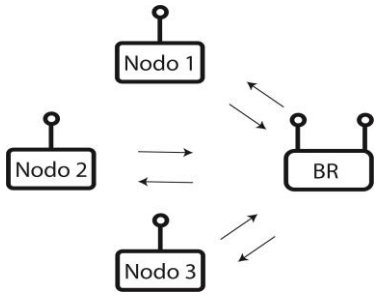


Figura 1: *Topología en estrella.*

Topología en malla-híbrida: la información se puede enviar y a su vez recibir al mismo tiempo, dando como resultado que la estructura de la red sea extensa siendo esto una gran ventaja, puesto que cada nodo tiene diferentes vías para el envío y recepción de información, en este caso si un nodo falla automáticamente la red lo reconfigura para lograr otra conexión. En la Figura 2 se observa la topología en forma de malla [5].

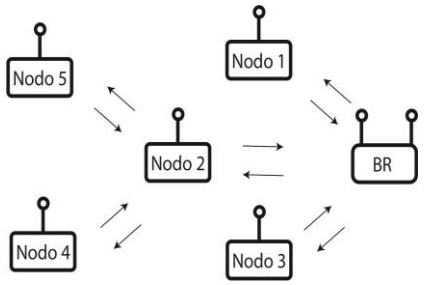


Figura 2: *Topología en malla-híbrida.*

C. Sistema Operativo

Contiki es un sistema operativo escrito en lenguaje C, Java y Python, utilizado en sistemas embebidos orientado a redes IoT, fue desarrollado para pequeños sistemas integrados de baja potencia, entre ellos nodos de redes de sensores, además de ser un S.O. ligero y de código abierto, es compatible con protocolos IPv4 e IPv6 en conjunto con estándares de bajo consumo de anchos de banda como 6LowPAN, RPL, COAP. Contiki además posee un software llamado Cooja para realizar simulaciones [6].

Contiki cuenta con una implementación en una máquina virtual denominada Instant Contiki la cual fue diseñada para su ejecución inmediata en conjunto con todas las herramientas, tanto en simuladores como en compiladores para las

diferentes plataformas de hardware como por ejemplo RE-MOTE, Launchpad, Z1, Sky, etc.

D. Enrutamiento

Contiki utiliza por defecto el protocolo RPL, mismo que provee enrutamiento para redes de baja potencia y con pérdidas denominado LLN (Low Power and Lossy Networks), RPL es un protocolo de vector distancia proactivo, lo que da como resultado el trazado de rutas tan rápido como converja la red informando a sus nodos contiguos los diferentes cambios dentro de la topología existente.

RPL define una topología en árbol DAG (Gráfica Acíclica Dirigida), que parte desde el nodo raíz (DODAG) y cuya función es elegir la mejor ruta en el menor rango posible; existen tres tipos de mensajes: [7].

DIS. - Solicita información desde el nodo raíz (DODAG) para descubrir redes existentes.

DIO. - Envía la información del DAG como respuesta de un DIS, este mensaje se envía cada cierto tiempo para la actualización de los cambios que existan en la red.

DAO. -Envía la información que permite a un nodo descubrir quien está directamente conectado al DAG (Nodo padre).

E. Servidores MQTT

MQTT (Message Qeue Telemetry Trasnport) es un protocolo de capa aplicación tanto de publicación como suscripción del servidor al cliente estandarizado por la OASIS (Organization for the Advancement of Structured Information Standards), se destaca por ser liviano y simple, diseñado para su fácil implementación, haciéndolo ideal para una comunicación máquina a máquina (M2M) e "Internet de las cosas" [8].

Al ser de publicación/suscripción el servidor o Broker MQTT actúa en conjunto puesto que es el encargado de la gestión de la red para transmitir mensajes MQTT entre sus clientes [9].

Eclipse Mosquito es un bróker o servidor MQTT privado de código abierto licenciado por la EPL (Eclipse Public License), que se caracteriza por ser liviano y bajo de recursos haciéndolo óptimo en aplicaciones de IoT basadas en hardware y sensores con baja capacidad de procesamiento [10].

Para la implementación del broker MQTT existen un número de puertos disponibles dependiendo si es público o privado, en este caso se utiliza el puerto 1883 ya que se envían y reciben mensajes MQTT sin cifrar a diferencia de los demás puertos que son cifrados [11].

Desde la página oficial de MQTT [12] se puede descargar el repositorio para diferentes plataformas, una de ellas para el sistema operativo Raspbian, mismo que se utiliza en la implementación del prototipo Smart Home propuesto.

Eclipse Paho es un software de código abierto orientado a protocolos de mensajería MQTT para aplicaciones dirigidas a internet de las cosas como también M2M, contiene implementaciones de cliente tanto en publicación como en suscripción según la necesidad del usuario, proveyendo APIs asíncronas y síncronas, por una lado las APIs asíncronas facilitan al programador el control de sus clientes suscritos, respecto a las APIs síncronas, estás ayudan a implementar una lógica en las aplicaciones [13].

F. Plataformas de desarrollo

Para la realización del prototipo se utiliza el ordenador Raspberry PI 3, por sus características que se detallan en la Tabla 3 [6].

Raspberry Pi es un ordenador de bajo costo, su tamaño reducido y sus puertos GPIO (entrada/salida), que lo hacen ideal para el desarrollo de aplicaciones para IoT, gracias a que permite el control de componentes electrónicos y está basado en el sistema operativo Raspbian.

Tabla 3. *Características de Raspberry Pi 3.*

Procesador	Quad Core BCM 2837 (64 bits)
Memoria RAM	1.2GB
Frecuencia de reloj	1.2 GHz
Puertos	4 USB 2.0/ 1 HDMI/ microSD
Pines	40 GPIO
Interfaz	Bluetooth 4.1/ Fast Ethernet/ WLAN IEEE 802.11.b/g/n

Adicionalmente para el desarrollo del prototipose usará la plataforma RE-MOTE de Zolertia, mismo que cuenta con un Sistema sobre un chip CC2538 y un transceptor CC1200 de Texas Instruments, basado en el ARM Cortex-M3 a 32MHz desarrollado para redes WSN [14].

G. Plataforma IoT

Ubidots es una plataforma desarrollada para IoT cuya función es facilitar la creación de prototipos para monitorear, controlar y automatizar de una manera remota los procesos enviados a la nube con el fin de tomar acciones o alertar al usuario de los eventos que ocurren en tiempo real, entre sus aplicaciones desarrolladas destacan los campos de: salud, energía, servicios públicos, transporte, edificios, casas inteligentes, entre otros [15].

H. Rastreador de paquetes (Sniffer)

Un Sniffer permite capturar los paquetes transmitidos para analizarlos con detalle. Una herramienta popular es Wireshark, sin embargo, se necesita hardware adicional para que se pueda capturar paquetes inalámbricos encapsulados bajo el estándar 802.15.4. Ejemplos de aplicaciones comerciales con funcionalidades similares son SmartRF de Texas Intruments o Sensniff [16]. Para la implantación de este prototipo se usa Sensniff en una RE-MOTE y Wireshark.

III. DISEÑO E IMPLEMENTACIÓN

A. RE-MOTE como nodo Gateway

Para la instalación de Instant Contiki se usó el software de virtualización VMware Workstation player en conjunto con la máquina virtual respectiva.

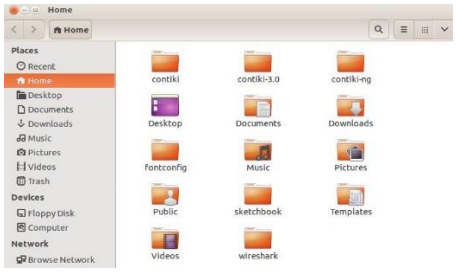


Figura 3: *Instant Contiki.*

Contiki contiene una aplicación de enrutador de borde lista llamada border-router, la cual se encuentra en: Contiki/examples/ipv6/rpl-border-router. Una vez conectado un nodo RE-MOTE se usó el comando:

make TARGET=zoul BOARD=remote-revb border-router.upload

Para la ejecución border router:

Contiki/examples/ipv6/rpl-border-router con el siguiente comando:

make connect-router

```
ifconfig tun0 add fd00::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

Server IPv6 addresses:
 fd00::212:4b00:1932:e305
 fe80::212:4b00:1932:e305
```

Figura 4: *Aplicación border router.*

B. RE-MOTE como nodo

De la misma manera se conectó un nodo RE-MOTE mediante el cable USB a la máquina virtual, en la dirección:

/Contiki/examples/Nodo smarthome/

Para cargar el código desarrollado se utilizó el siguiente comando:

make nodo.upload

Una vez ejecutado el Gateway los vecinos, en este caso los nodos sensores se enlazaron a él como muestra la Figura 5.

Figura 5: *Servidor web Contiki-RPL.*

C. RE-MOTE como Sniffer

Se compiló el archivo sensniff.c localizado en Contiki/examples/sensniff en una RE-mote con el siguiente comando:

make TARGET=zoul BOARD=remote-revb sensniff.upload

Además, se ejecutó el script sensniff.py mediante Python localizado en Contiki/tools/sensniff con el siguiente comando:

python sensniff.py -d /dev/ttyUSB1,

Creando así una interfaz la cual puede ser compatible con Wireshark cuyo directorio de archivos se encuentra en /tmp/sensniff, la cual debe ser escrita manualmente como muestra la Figura 6.

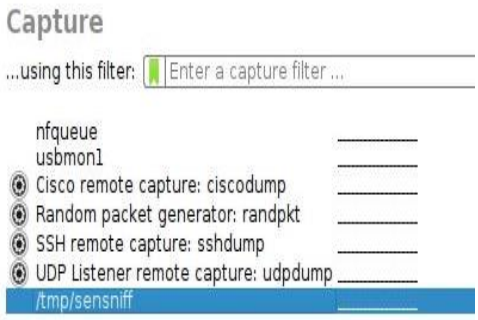


Figura 6: *Interfaz Sensniff-Wireshark.*

D. Prototipo

El prototipo implementado se basa en una red 6LoWPAN en forma de estrella Figura 7, es decir cada nodo o mota se conecta directamente al enrutador de borde (border router), el mismo que cuenta con dos nodos sensores y un nodo Gateway conectado a Ubidots para la interacción del usuario desde la nube.

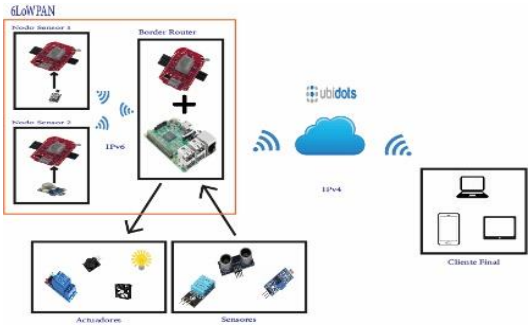


Figura 7: *Prototipo en topología estrella.*

En la Figura 8 se muestra una red en forma de malla es decir el nodo puede convertirse en padre (DODAG) capturando la información del nodo más lejano y enviándola al router de borde.

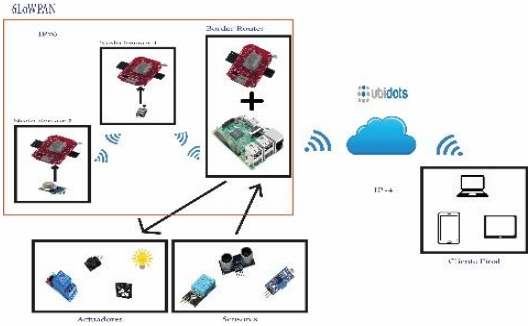


Figura 8: *Prototipo en topología malla.*

En la Figura 9 se puede observar la comunicación en malla-híbrida de 3 nodos uno más lejano del otro con respecto al router de borde, es decir si el rango de cobertura de un nodo no es suficiente para alcanzar al router de frontera, este usa a un nodo vecino que si lo haga para transmitir su información.

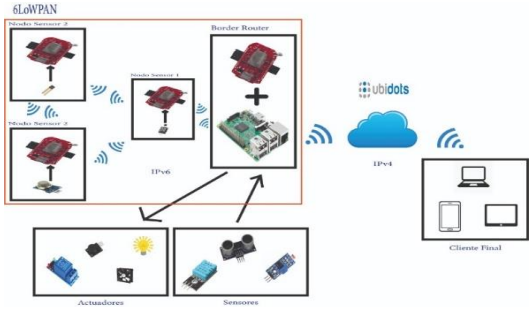


Figura 9: *Prototipo en topología híbrida.*

IV. ANÁLISIS Y RESULTADOS

Topología estrella: en una configuración estrella todos los nodos cercanos se conectan directamente al nodo enrutador o Gateway esto es apreciable en la Figura 10.

Figura 10: *Servidor Web RPL con nodos sensores en estrella.*

Como puede observarse el código compilado del enrutador de borde incorpora una dirección de ipv6 fd00::212:4b00:192:e305, dicha dirección se vincula a un servidor web el cual muestra todos los nodos conectados a él también conocidos como vecinos, además de las rutas que ellos usan para conectarse.

En una topología estrella los nodos no utilizan más de una ruta para conectarse al enrutador, por ejemplo, el nodo fd00::2124b00:14d5:2db0/128 se conecta vía fe80::2124b00:14d5:2db0/128

Se puede verificar también haciendo un ping hacia cualquier nodo en este caso a fd00::2124b00:14d5:2db0 como se muestra en la Figura 11.

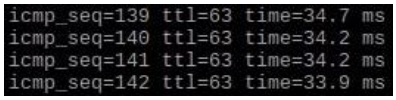


Figura 11: *Pruebas de conectividad a nodo sensor en topología estrella.*

Como se puede apreciar en la figura 11 el límite de saltos TTL es 63 es decir que el paquete ICMPv6

atravesó solamente un enrutador hasta alcanzar el nodo entonces su TTL inicial debió ser 64.

Topología malla-híbrida: en una configuración malla los nodos en la red 6LowPan no necesariamente deben conectarse directamente al router de borde para empezar a transmitir, es decir si un nodo se encuentra demasiado lejos utiliza a un nodo vecino relativamente cerca a ambos y así transmitir su información, como se muestra en la Figura 12.

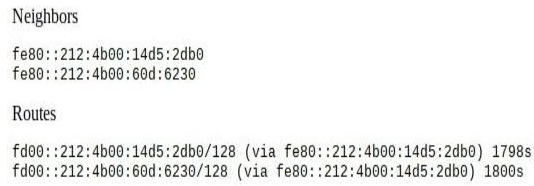


Figura 12: *Servidor Web RPL con nodos sensores en malla-hibrida.*

Como puede observarse en el apartado de rutas el nodo cuya dirección ipv6 es fd00::212:4b00:60d:6230 se conecta al enrutador por medio del nodo fe80::2124b00:14d5:2db0.

De la misma manera se puede verificar observando el número de saltos que tiene un paquete ICMPv6 hacia el mismo nodo sensor cuyo TTL disminuyó en dos es decir 62 como se muestra en la Figura 13.

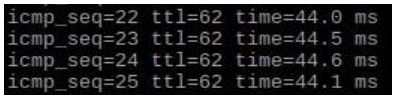


Figura 13: *Pruebas de conectividad a nodo sensor en topología malla.*

Implementación de red 6LowPan

Para la implementación del prototipo propuesto se optó por la topología de la figura 7 y cuya descripción se muestra en la Tabla 4.

Tabla 4: Función de cada nodo

NODO	DIRECCIÓN	FUNCIÓN
Nodo 1	fd00::212:4b00:192:e305 fd00::1	Router de borde
Nodo 2	fd00::2124b00:14d5:2db0	Sensor humo
Nodo 3	fd00::2124b00:60d:6230	Sensor temperatura
Nodo 4	NA	Sniffer

Para comprobar que los paquetes 6LowPan son enviados mediante 802.15.4 se usa la herramienta Wireshark junto con la implementación de un sniffer compilado en Contiki y ejecutado con Python cuyo resultado se aprecia en la Figura14.

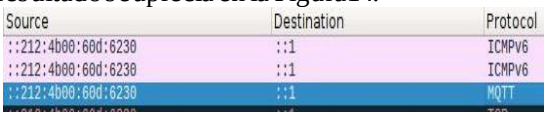


Figura 14: *Paquetes 6LowPan capturados mediante Wireshark.*

En la Figura 15 se muestra la pila de protocolos del paquete capturado, desde la capa física, la capa enlace, la capa de adaptación, la capa de red, bajo el estándar 802.15.4 tal como se muestra en la Figura 15.

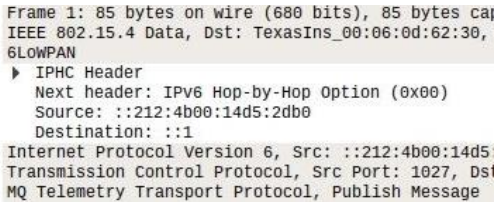


Figura 15: *Paquete 6LowPan capturado.*

Parámetros del prototipo

Para demostrar la funcionalidad del prototipo de Smart home, se ha previsto un escenario que contenga los siguientes ambientes:

- Cocina
- Baño
- Habitación 1
- Habitación 2
- Sala
- Entrada

En cada ambiente se realizan mediciones para determinar los valores de cobertura, latencia y TTL como se muestra en la Tabla5:

Tabla 5: *Parámetros obtenidos del Prototipo.*

AMBIENTE	POTENCIA [dB]	DISTANCIA [m]	LATENCIA [s]
Habitación 1	-76	7.8	10
Habitación 2	-84	11	10
Cocina	-86	10	10
Entrada	-78	10	9.80
Sala	-84	13	10
Baño	-67	4	10
Estudio	-67	2	10
Fuera de alcance	-92	15	NA

Los parámetros de potencia fueron tomados del promedio de 10 paquetes capturados con la herramienta Wireshark, este dato puede ser encontrado en el detalle de la capa enlace como se muestra en la Figura 16.

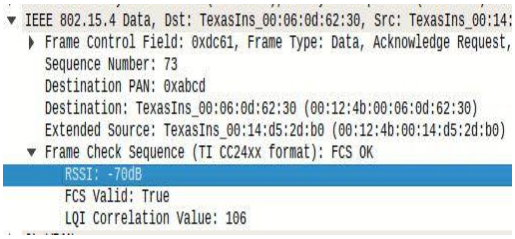


Figura 16: *RSSI de paquete 6LowPan capturado.*

La distancia se mide partiendo desde la ubicación del nodo Gateway hacia la ubicación del nodo sensor como se muestra en la tabla 5, al final se obtiene la siguiente Figura 17.

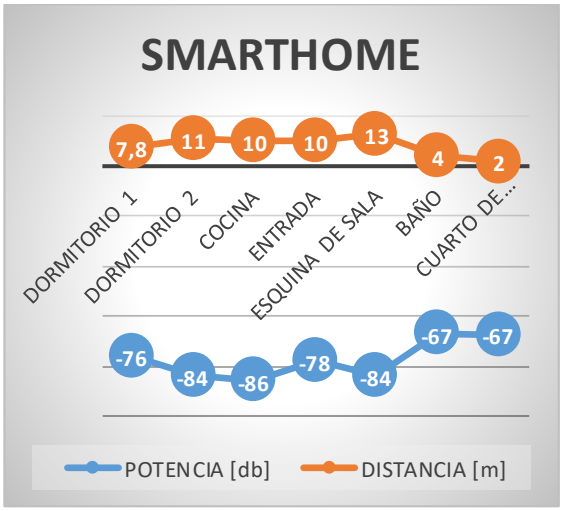


Figura 17: *Relación potencia y distancia.*

De acuerdo con la gráfica se puede notar que el nodo más alejado del Gateway es el que se encuentra en una esquina de la sala a una distancia de 13 metros con una potencia de -84dB, por otro lado el nodo más cercano se encuentra a 2 metros con una potencia de -67 dB, sin embargo esta potencia suele mejorar hasta -45 dB, en estas mediciones se tuvo una latencia de 10 segundos aproximadamente, esto tomando en cuenta el tiempo de captura de 10 paquetes MQTT y promediándolos como muestra la tabla 5.

Puesta en marcha del prototipo Smart Home

El prototipo Smart Home utiliza nodos sensores y un Gateway que se conectan usando al protocolo 6lowpan, sin embargo, el envío de datos desde estos sensores (dióxido de carbono y temperatura) utiliza el protocolo MQTT, el cual publica dicha información en la consola del ordenador Raspberry con ayuda de los scripts Python. Posteriormente estos datos son enviados al internet a través de la plataforma Ubidots donde es mostrada al usuario para la ejecución de determinadas acciones en caso de ser necesarias.

Figura 18: *Panel de visualización de Ubidots.*

V. CONCLUSIONES

El protocolo 6LowPan es eficiente en topologías de tipo estrella, malla o hibrida gracias a RPL que permite conectarse de forma directa al nodo Gateway o en su defecto cuando se usa un nodo sensor mediante un enlace transparente.

La implementación de una WSN con 6LowPan explota la capa de adaptación entre la capa de enlace y red usando el parámetro IPHC, mismo que comprime una dirección IPv6 para que pueda ser trasmitida dentro de los parámetros de 802.15.4.

En la implementación del prototipo, el alcance máximo entre un sensor y el Gateway fue de 13 metros a una potencia de
-84dB y con una latencia de 10s, y por otra parte la potencia optima de transmisión fue de -67 dB de 2 a 4 metros lo cual determina que a mayor distancia menor es la potencia de transmisión.

Al momento de realizar pruebas de conectividad se observó que el TTL o límite de salto a un nodo cercano es 63 mientras que de un nodo lejano es de 62 lo que determina el número de saltos que les toma a los paquetes para llegar a su destino.

La implementación del prototipo de una Smart Home con nodos sensores conectados a través de 6LowPan hacia una plataforma IoT permite obtener notificaciones en tiempo real para que los usuarios puedan tomar decisiones oportunamente.

El prototipo planteado es completamente escalable puesto que no solo se aplica para sensores de humo o temperatura, sino también es posible la

integración de cualquier tipo de sensores que soporten 6LowPan y otro tipo de protocolos gracias a Python y librerías específicas.

VI. RECOMENDACIONES

Para mayor seguridad en el envío mensajes MQTT, se propone implementar una estrategia de encriptación utilizando los puertos seguros como el 8883.

Para efectos de establecer un análisis comparativo en cuanto al comportamiento de los sensores se sugiere realizar un prototipo con sensores digitales pues estos consumen menos energía que sensores analógicos.

Para una eficiente implementación de este prototipo se sugiere el uso del hardware RE-MOTE de Zolertia debido a su compatibilidad con Contiki teniendo en cuenta que no es así con el simulador Cooja.

VII. REFERENCIAS

[1] PC Componentes y Multimedia SLU, «Qué es Smarthome | Pccomponentes,» [En línea]. Available: https://www.pccomponentes.com/que-es-smarthome. [Último acceso: 05 09 2019].

[2] T. Pinillos y F. Edgar, IP version 6 : la nueva generacion IP, 2 ed., vol. 2, Zulia: Telematique, 2003, pp. 50-57.

[3] A. Rayes y S. Salam, Internet of Things From Hype to Reality, 2 ed., Cham: Springer International Publishing, 2017, p. 303.

[4] A. Ñ. M. Siamara, «Desarrollo de un prototipo para la automatización de un sistema de riego de agua y control remoto mediante la plataforma Zolertia Remote,» Escuela Politecnica Nacional, 2019.

[5] S. R. M. Cantillo, «Desarrollo de aplicaciones basadas en WSN,» Valencia, 2010.

[6] Ñ. M. S. Adriana, «Desarrollo de un prototipo para la automatización de un sistema de riego de agua y control remoto mediante la plataforma Zolertia RE-Mote,» Quito, 2019.

[7] Mugdhe y S. Pai., «Network Stack,» Anrg.usc.edu, 2019. [En línea]. Available: https://anrg.usc.edu/contiki/index.php/Network_Stack. [Último acceso: 23 10 2019].

[8] Docs.oasis-open.org, «MQTT Veersion 3.1.1,» 2019. [En línea]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf. [Último acceso: 12 12 2019].

[9] Mqtt.org, «FAQ - Frecuently Asked Questions | MQTT,» 2019, [En línea]. Available: http://mqtt.org/faq. [Último acceso: 12 12 2019].

[10] The Journal of Open Source Software, «Mosquitto: server and client implementation of the MQTT protocol,» 2017. [En línea]. Available: https://www.theoj.org/joss-papers/joss.00265/10.21105.joss.00265.pdf. [Último acceso: 4 11 2019].

[11] «test.mosquitto.org,» [En línea]. Available: http://test.mosquitto.org/. [Último acceso: 04 11 2017].

[12] «Eclipse Mosquitto,» Eclipse Mosquitto, 08 01 2018. [En línea]. Available: https://mosquitto.org/. [Último acceso: 04 11 2019].

[13] W.-J. Chen, R. Gupta, V. Lampkin, D. M. Robertson, N. Subrahmanyam y I.B.M, Responsive Mobile User Experience Using MQTT and IBM MessageSight., IBM Redbooks, 2014, p. 118.

[14] A. L. Colina, A. Vives, A. Bagula, M. Zennaro y E. Piertrosemoli, Internet de las Cosas, 2015.

[15] «IoT platform | Internet of Things | Ubidots,» 12 12 2019. [En línea]. Available: https://ubidots.com/. [Último acceso: 12 12 2019].

[16] herjulf, «https://github.com/,» 31 marzo 2017. [En línea]. Available: https://github.com/contiki-os/contiki/tree/master/examples/sensniff. [Último acceso: 12 diciembre 2019].